# WEAPONIZING 1 DAY VULNS

# ABOUT ME – KOREY YOUNG

- Bloomsburg University – Computer Science.
- But I have always had an interest in cyber security and hacking / red team activities
  - Finding ways to make computers do things that they were not meant to do;
  - So that we can get it fixed before the bad guys discover it and try to use it.
- Dakota State University - Masters in Information Assurance
- SANS GWAPT, Sec 642, GPEN, GXPN, Sec 760
- My day job is penetration testing for an IT company, trying to hack into our applications.
- In my free time after work and on weekends, I do cyber security and exploit research.

# 1 DAY VULNERABILITY

- Vulnerabilities are called zero-days during the time between disclosure (when the vulnerability is revealed) and when it is patched
- Once the vulnerability has been patched, it is called a 1-day or n-days vulnerability

# PATCH DIFFING

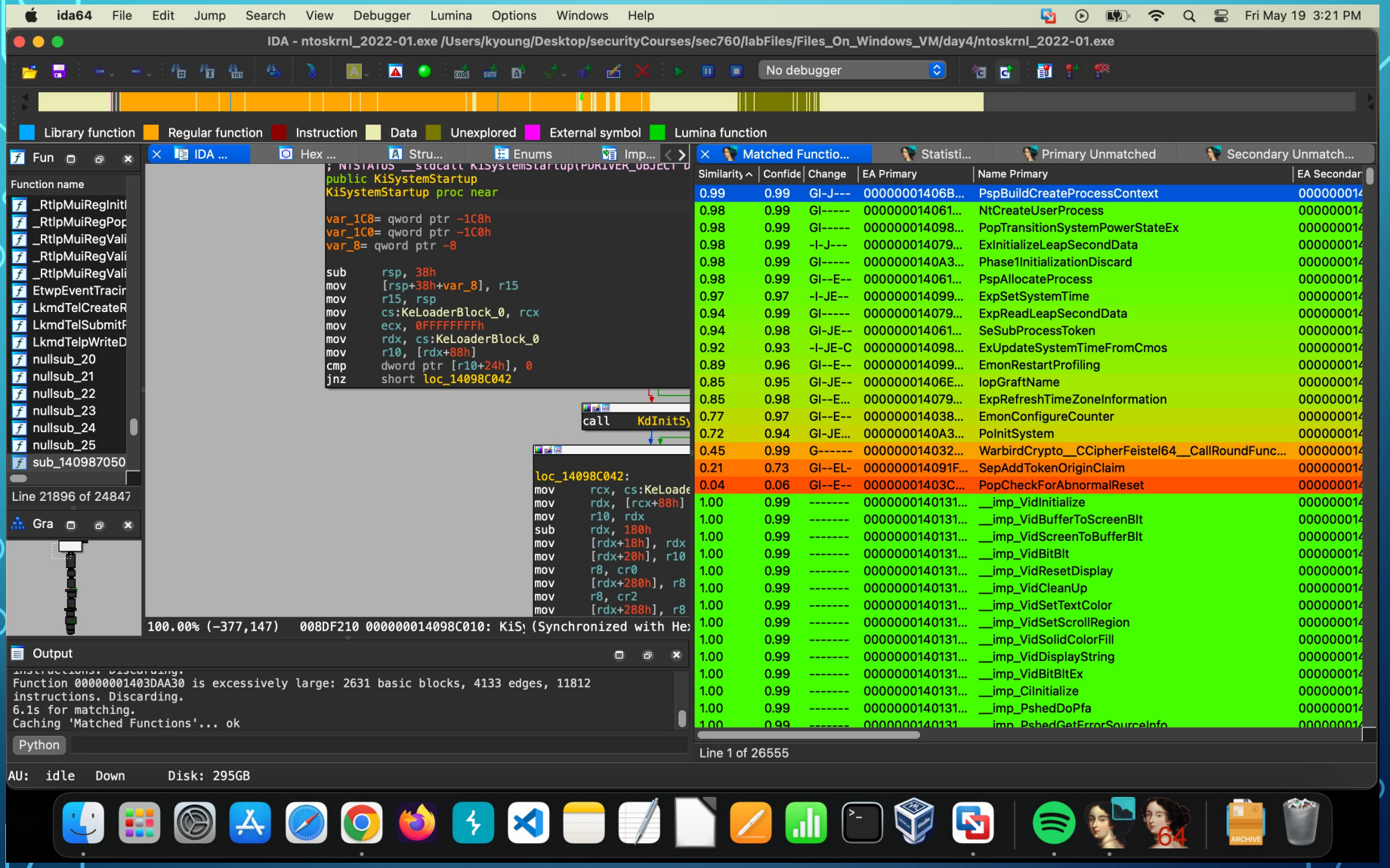
- How do reverse engineers find what changed in patched software, in order to exploit the weakness?
- They use Patch Diffing, which is when you use a tool to examine the contents of 2 binary files, to see what changed between them

# PATCH DIFFING

- Hex Rays' BinDiff is one of the most well known tools to use to do this
  - Also Radiff2 from Radare and Diaphora
- You can manually import the resulting diff files into a decompiler such as IDAPro or Ghidra
  - Or IDAPro professional does the diffing via a menu option, via a plugin with BinDiff

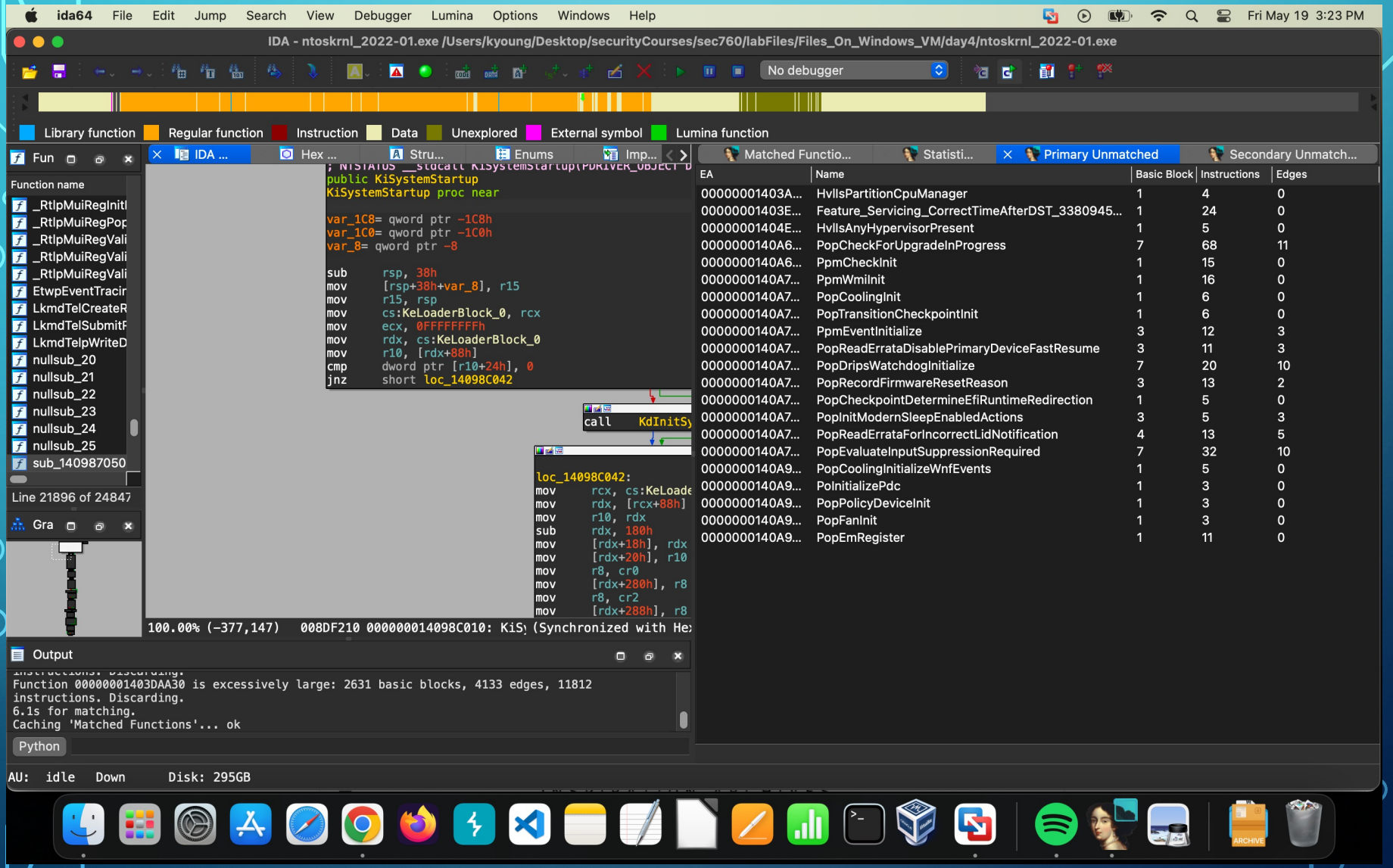
# PATCH DIFFING

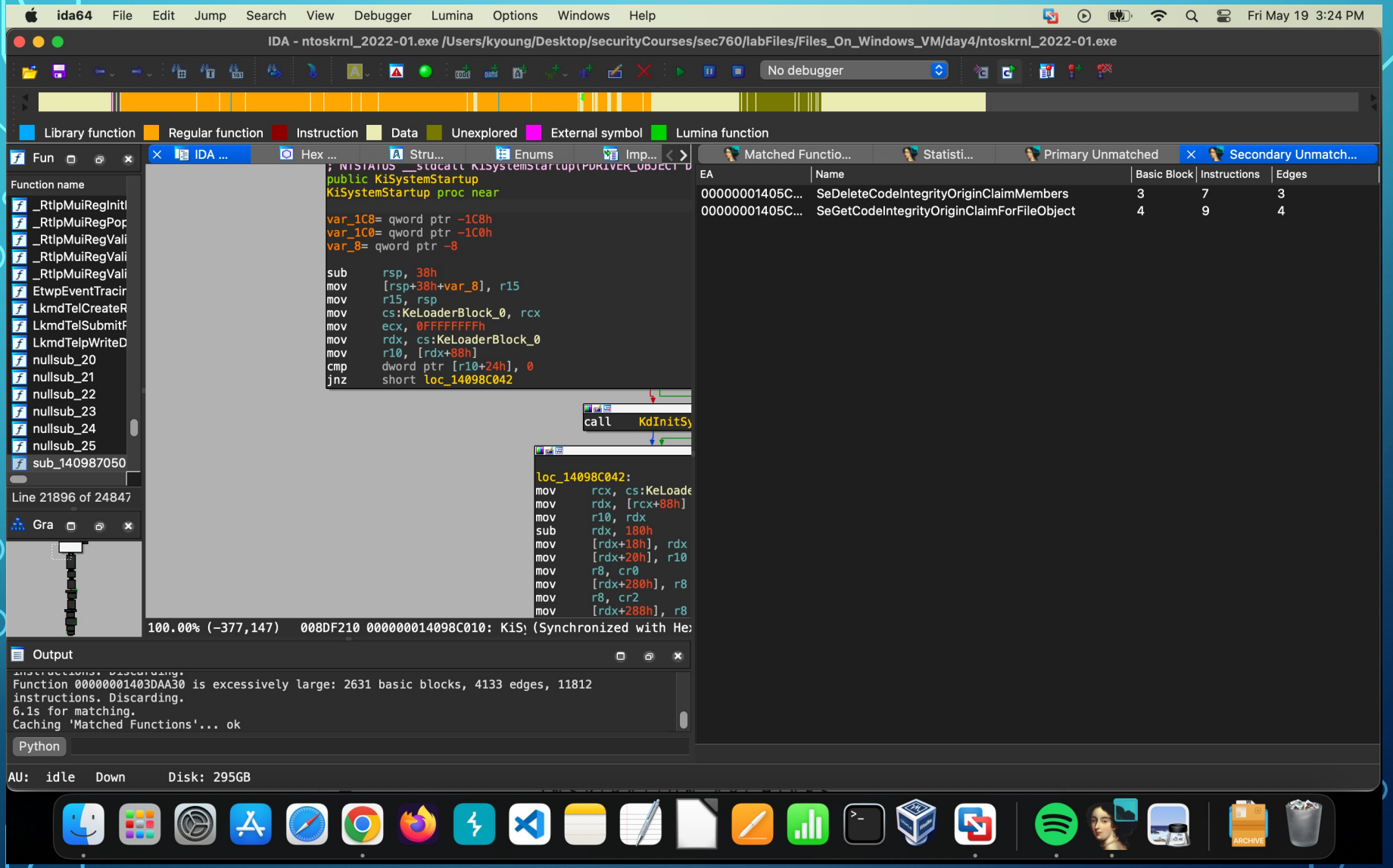
- Red = more code changes
- Green = less code changes


ida64 File Edit Jump Search View Debugger Lumina Options Windows Help
Fri May 19 3:21 PM
IDA - ntoskrnl_2022-01.exe /Users/kyoung/Desktop/securityCourses/sec760/labFiles/Files_On_Windows_VM/day4/ntoskrnl_2022-01.exe
No debugger
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Lumina function
Fun
Function name
_RtlpMuiRegInitl
_RtlpMuiRegPop
_RtlpMuiRegVali
_RtlpMuiRegVali
_RtlpMuiRegVali
EtwpEventTracir
LkmdTelCreateR
LkmdTelSubmitF
LkmdTelpWriteD
nullsub_20
nullsub_21
nullsub_22
nullsub_23
nullsub_24
nullsub_25
sub_140987050
Line 21896 of 24847
Gra
IDA ...
Hex ...
Stru...
Enums
Imp...
public KiSystemStartup
KiSystemStartup proc near
var_1C8= qword ptr -1C8h
var_1C0= qword ptr -1C0h
var_8= qword ptr -8
sub rsp, 38h
mov [rsp+38h+var_8], r15
mov r15, rsp
mov cs:KeLoaderBlock_0, rcx
mov ecx, 0FFFFFFFFh
mov rdx, cs:KeLoaderBlock_0
mov r10, [rdx+88h]
cmp dword ptr [r10+24h], 0
jnz short loc_14098C042
call KdInitSy
loc_14098C042:
mov rcx, cs:KeLoade
mov rdx, [rcx+88h]
mov r10, rdx
sub rdx, 180h
mov [rdx+18h], rdx
mov [rdx+20h], r10
mov r8, cr0
mov [rdx+280h], r8
mov r8, cr2
mov [rdx+288h], r8
100.00% (-377,147) 008DF210 000000014098C010: KiSy (Synchronized with He
Matched Functio...
Statisti...
Primary Unmatched
Secondary Unmatch...
Similarity
Confide
Change
EA Primary
Name Primary
EA Secondar
0.99 0.99 GI-J--- 00000001406B... PspBuildCreateProcessContext
0.98 0.99 GI----- 000000014061... NtCreateUserProcess
0.98 0.99 GI----- 000000014098... PopTransitionSystemPowerStateEx
0.98 0.99 -I-J--- 000000014079... ExInitializeLeapSecondData
0.98 0.99 GI----- 0000000140A3... Phase1InitializationDiscard
0.98 0.99 GI--E-- 000000014061... PspAllocateProcess
0.97 0.97 -I-JE-- 000000014099... ExpSetSystemTime
0.94 0.99 GI----- 000000014079... ExpReadLeapSecondData
0.94 0.98 GI-JE-- 000000014061... SeSubProcessToken
0.92 0.93 -I-JE-C 000000014098... ExUpdateSystemTimeFromCmos
0.89 0.96 GI--E-- 000000014099... EmonRestartProfiling
0.85 0.95 GI-JE-- 00000001406E... IopGraftName
0.85 0.98 GI--E... 000000014079... ExpRefreshTimeZoneInformation
0.77 0.97 GI--E-- 000000014038... EmonConfigureCounter
0.72 0.94 GI-JE... 0000000140A3... PoInitSystem
0.45 0.99 G------ 000000014032... WarbirdCrypto__CCipherFeistel64__CallRoundFunc...
0.21 0.73 GI--EL- 000000014091F... SepAddTokenOriginClaim
0.04 0.06 GI--E-- 00000001403C... PopCheckForAbnormalReset
1.00 0.99 ------- 0000000140131... __imp_VidInitialize
1.00 0.99 ------- 0000000140131... __imp_VidBufferToScreenBlt
1.00 0.99 ------- 0000000140131... __imp_VidScreenToBufferBlt
1.00 0.99 ------- 0000000140131... __imp_VidBitBlt
1.00 0.99 ------- 0000000140131... __imp_VidResetDisplay
1.00 0.99 ------- 0000000140131... __imp_VidCleanUp
1.00 0.99 ------- 0000000140131... __imp_VidSetTextColor
1.00 0.99 ------- 0000000140131... __imp_VidSetScrollRegion
1.00 0.99 ------- 0000000140131... __imp_VidSolidColorFill
1.00 0.99 ------- 0000000140131... __imp_VidDisplayString
1.00 0.99 ------- 0000000140131... __imp_VidBitBltEx
1.00 0.99 ------- 0000000140131... __imp_CiInitialize
1.00 0.99 ------- 0000000140131... __imp_PshedDoPfa
Line 1 of 26555
Output
Function 00000001403DAA30 is excessively large: 2631 basic blocks, 4133 edges, 11812
instructions. Discarding.
6.1s for matching.
Caching 'Matched Functions'... ok
Python
AU: idle Down Disk: 295GB

# PATCH DIFFING

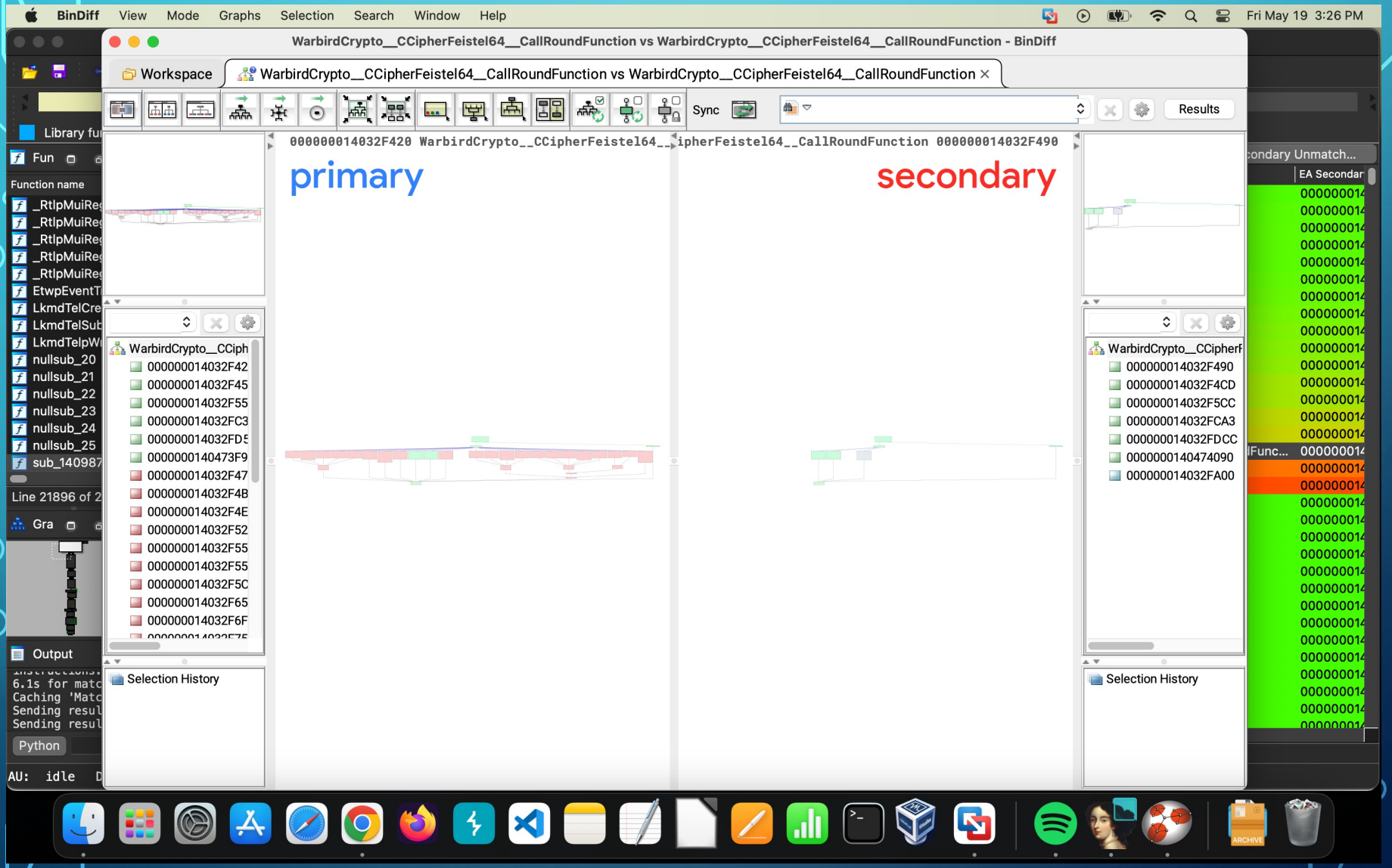
- Primary Unmatched = functions only found in the first of the diffed files
- Secondary Unmatched = functions only found in the second of the diffed files

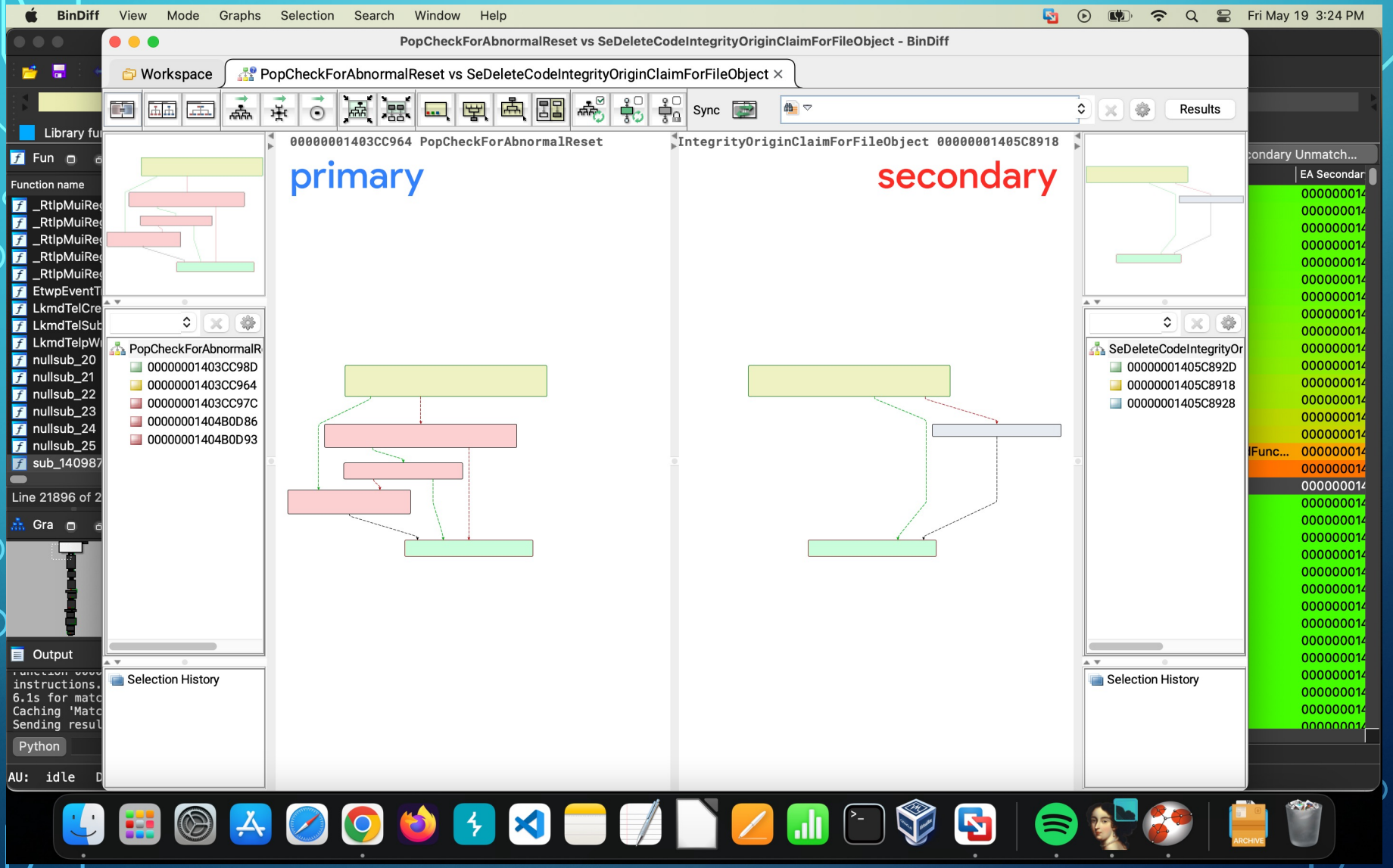

ida64 File Edit Jump Search View Debugger Lumina Options Windows Help
Fri May 19 3:23 PM
IDA - ntoskrnl_2022-01.exe /Users/kyoung/Desktop/securityCourses/sec760/labFiles/Files_On_Windows_VM/day4/ntoskrnl_2022-01.exe
No debugger
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Lumina function
Fun
Function name
_RtlpMuiRegInitl
_RtlpMuiRegPop
_RtlpMuiRegVali
_RtlpMuiRegVali
_RtlpMuiRegVali
EtwpEventTracir
LkmdTelCreateR
LkmdTelSubmitF
LkmdTelpWriteD
nullsub_20
nullsub_21
nullsub_22
nullsub_23
nullsub_24
nullsub_25
sub_140987050
Line 21896 of 24847
Gra
IDA ...
Hex ...
Stru...
Enums
Imp...
public KiSystemStartup
KiSystemStartup proc near
var_1C8= qword ptr -1C8h
var_1C0= qword ptr -1C0h
var_8= qword ptr -8
sub rsp, 38h
mov [rsp+38h+var_8], r15
mov r15, rsp
mov cs:KeLoaderBlock_0, rcx
mov ecx, 0FFFFFFFFh
mov rdx, cs:KeLoaderBlock_0
mov r10, [rdx+88h]
cmp dword ptr [r10+24h], 0
jnz short loc_14098C042
call KdInitSy
loc_14098C042:
mov rcx, cs:KeLoade
mov rdx, [rcx+88h]
mov r10, rdx
sub rdx, 180h
mov [rdx+18h], rdx
mov [rdx+20h], r10
mov r8, cr0
mov [rdx+280h], r8
mov r8, cr2
mov [rdx+288h], r8
100.00% (-377,147) 008DF210 000000014098C010: KiSy (Synchronized with Hex
Matched Functio...
Statisti...
Primary Unmatched
Secondary Unmatch...
EA
Name
Basic Block
Instructions
Edges
00000001403A... HvIlsPartitionCpuManager 1 4 0
00000001403E... Feature_Servicing_CorrectTimeAfterDST_3380945... 1 24 0
00000001404E... HvIlsAnyHypervisorPresent 1 5 0
0000000140A6... PopCheckForUpgradeInProgress 7 68 11
0000000140A6... PpmCheckInit 1 15 0
0000000140A7... PpmWmiInit 1 16 0
0000000140A7... PopCoolingInit 1 6 0
0000000140A7... PopTransitionCheckpointInit 1 6 0
0000000140A7... PpmEventInitialize 3 12 3
0000000140A7... PopReadErrataDisablePrimaryDeviceFastResume 3 11 3
0000000140A7... PopDripsWatchdogInitialize 7 20 10
0000000140A7... PopRecordFirmwareResetReason 3 13 2
0000000140A7... PopCheckpointDetermineEfiRuntimeRedirection 1 5 0
0000000140A7... PopInitModernSleepEnabledActions 3 5 3
0000000140A7... PopReadErrataForIncorrectLidNotification 4 13 5
0000000140A7... PopEvaluateInputSuppressionRequired 7 32 10
0000000140A9... PopCoolingInitializeWnfEvents 1 5 0
0000000140A9... PoInitializePdc 1 3 0
0000000140A9... PopPolicyDeviceInit 1 3 0
0000000140A9... PopFanInit 1 3 0
0000000140A9... PopEmRegister 1 11 0
Output
Function 0000000140DAA30 is excessively large: 2631 basic blocks, 4133 edges, 11812
instructions. Discarding.
6.1s for matching.
Caching 'Matched Functions'... ok
Python
AU: idle Down Disk: 295GB

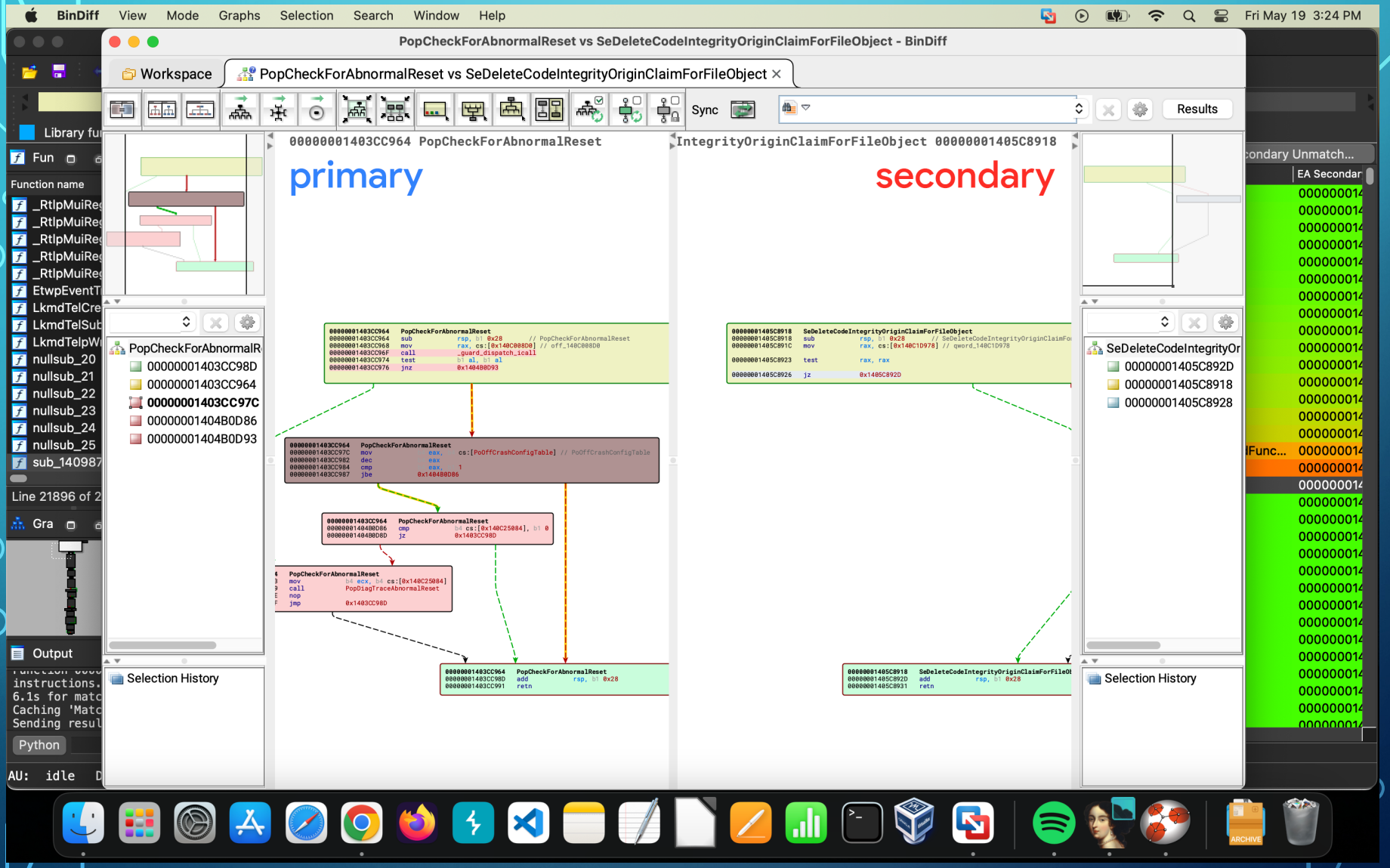

ida64 File Edit Jump Search View Debugger Lumina Options Windows Help
Fri May 19 3:24 PM
IDA - ntoskrnl_2022-01.exe /Users/kyoung/Desktop/securityCourses/sec760/labFiles/Files_On_Windows_VM/day4/ntoskrnl_2022-01.exe
No debugger
Library function
Regular function
Instruction
Data
Unexplored
External symbol
Lumina function
Fun
Function name
_RtlpMuiRegInitl
_RtlpMuiRegPop
_RtlpMuiRegVali
_RtlpMuiRegVali
_RtlpMuiRegVali
EtwpEventTracir
LkmdTelCreateR
LkmdTelSubmitF
LkmdTelpWriteD
nullsub_20
nullsub_21
nullsub_22
nullsub_23
nullsub_24
nullsub_25
sub_140987050
Line 21896 of 24847
Gra
IDA ...
Hex ...
Stru...
Enums
Imp...
public KiSystemStartup
KiSystemStartup proc near
var_1C8= qword ptr -1C8h
var_1C0= qword ptr -1C0h
var_8= qword ptr -8
sub rsp, 38h
mov [rsp+38h+var_8], r15
mov r15, rsp
mov cs:KeLoaderBlock_0, rcx
mov ecx, 0FFFFFFFFh
mov rdx, cs:KeLoaderBlock_0
mov r10, [rdx+88h]
cmp dword ptr [r10+24h], 0
jnz short loc_14098C042
call KdInitSy
loc_14098C042:
mov rcx, cs:KeLoade
mov rdx, [rcx+88h]
mov r10, rdx
sub rdx, 180h
mov [rdx+18h], rdx
mov [rdx+20h], r10
mov r8, cr0
mov [rdx+280h], r8
mov r8, cr2
mov [rdx+288h], r8
100.00% (-377,147) 008DF210 000000014098C010: KiSy (Synchronized with Hex
Matched Functio...
Statisti...
Primary Unmatched
Secondary Unmatch...
EA
Name
Basic Block
Instructions
Edges
00000001405C... SeDeleteCodeIntegrityOriginClaimMembers 3 7 3
00000001405C... SeGetCodeIntegrityOriginClaimForFileObject 4 9 4
Output
Function 0000001403DAA30 is excessively large: 2631 basic blocks, 4133 edges, 11812
instructions. Discarding.
6.1s for matching.
Caching 'Matched Functions'... ok
Python
AU: idle Down Disk: 295GB

# PATCH DIFFING

- You can drill down into the diff report, and view the added/removed/modified code sections
  - Green = added code sections
  - Red = removed code sections
  - Yellow = modified code sections


BinDiff View Mode Graphs Selection Search Window Help
Fri May 19 3:26 PM
WarbirdCrypto__CCipherFeistel64__CallRoundFunction vs WarbirdCrypto__CCipherFeistel64__CallRoundFunction - BinDiff
Workspace
WarbirdCrypto__CCipherFeistel64__CallRoundFunction vs WarbirdCrypto__CCipherFeistel64__CallRoundFunction
Sync
Results
000000014032F420 WarbirdCrypto__CCipherFeistel64__
ipherFeistel64__CallRoundFunction 000000014032F490
primary
secondary
WarbirdCrypto__CCiph
000000014032F42
000000014032F45
000000014032F55
000000014032FC3
000000014032FD5
0000000140473F9
000000014032F47
000000014032F4B
000000014032F4E
000000014032F52
000000014032F55
000000014032F55
000000014032F5C
000000014032F65
000000014032F6F
WarbirdCrypto__CCipherF
000000014032F490
000000014032F4CD
000000014032F5CC
000000014032FCA3
000000014032FDCC
0000000140474090
000000014032FA00
Selection History
Selection History
Function name
Output
Python
EA Secondar


BinDiff View Mode Graphs Selection Search Window Help
Fri May 19 3:24 PM
PopCheckForAbnormalReset vs SeDeleteCodeIntegrityOriginClaimForFileObject - BinDiff
Workspace
PopCheckForAbnormalReset vs SeDeleteCodeIntegrityOriginClaimForFileObject
Sync
Results
00000001403CC964 PopCheckForAbnormalReset
IntegrityOriginClaimForFileObject 00000001405C8918
primary
secondary
PopCheckForAbnormalR
00000001403CC98D
00000001403CC964
00000001403CC97C
00000001404B0D86
00000001404B0D93
SeDeleteCodeIntegrityOr
00000001405C892D
00000001405C8918
00000001405C8928
Selection History
Selection History
Function name
Output
Python
EA Secondar


BinDiff View Mode Graphs Selection Search Window Help
Fri May 19 3:24 PM
PopCheckForAbnormalReset vs SeDeleteCodeIntegrityOriginClaimForFileObject - BinDiff
Workspace
PopCheckForAbnormalReset vs SeDeleteCodeIntegrityOriginClaimForFileObject
Sync
Results
00000001403CC964 PopCheckForAbnormalReset
primary
IntegrityOriginClaimForFileObject 00000001405C8918
secondary
PopCheckForAbnormalR
00000001403CC98D
00000001403CC964
00000001403CC97C
00000001404B0D86
00000001404B0D93
SeDeleteCodeIntegrityOr
00000001405C892D
00000001405C8918
00000001405C8928
Selection History
Selection History
Output
Python
AU: idle

# VULNERABLE CODE

- The patched/fixed code could be protecting from a variety of attack types
  - Sanitizing input for safer usage
  - Handling data in a safer way (such as not overwriting memory)
  - Etc
- The data could be processed by the program immediately at the patched spot, or later on in the code

# VULNERABLE CODE

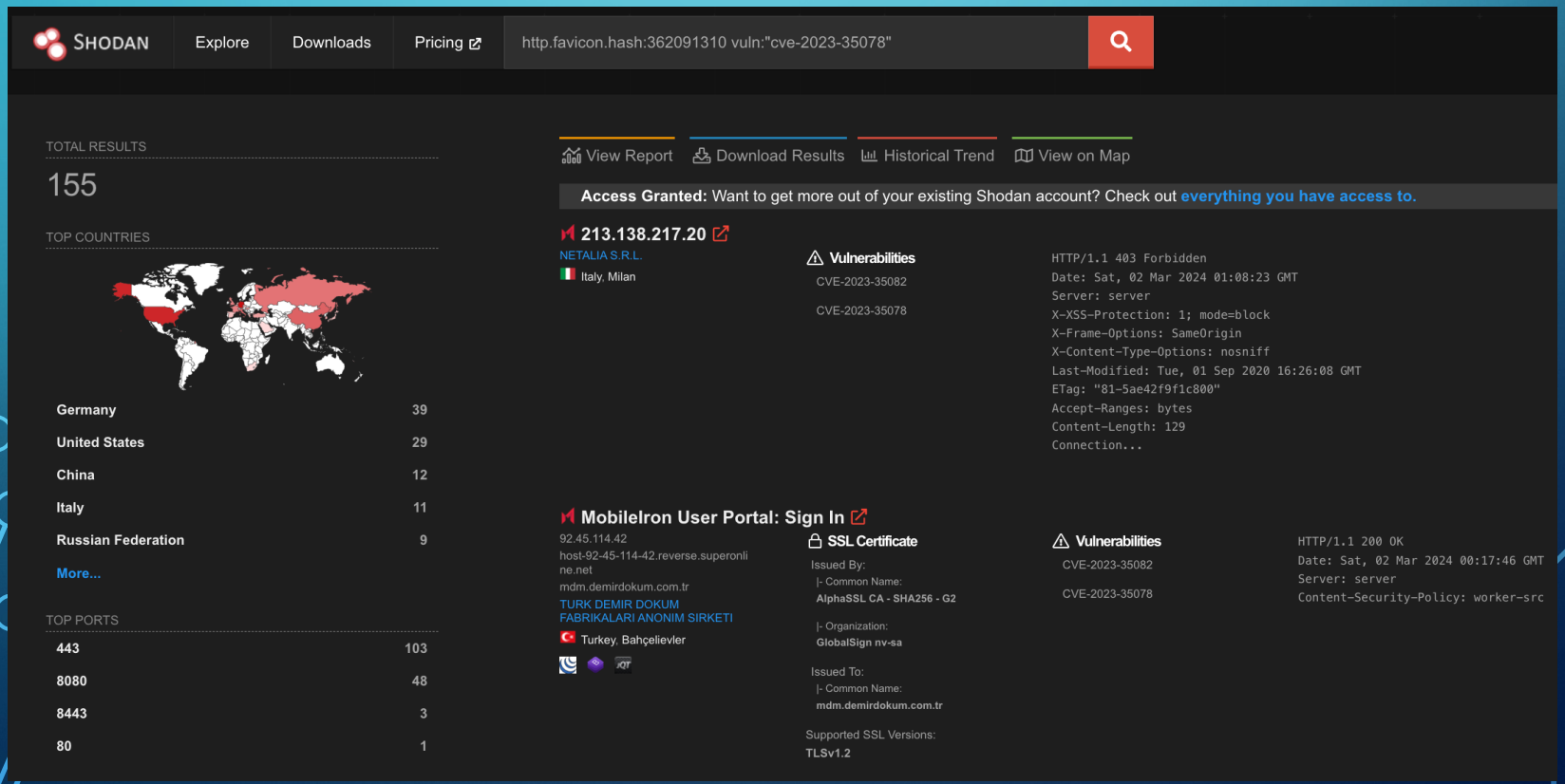
- Once you know what safeguards were lacking, you can use dangerous input to exploit any instances of the software that have not been patched yet

# PATCH LAG

- Unfortunately, many organizations take a long time to patch, leaving those systems vulnerable to attack
  - Patches can sometimes break systems, making organizations afraid to apply them
  - Or just laziness

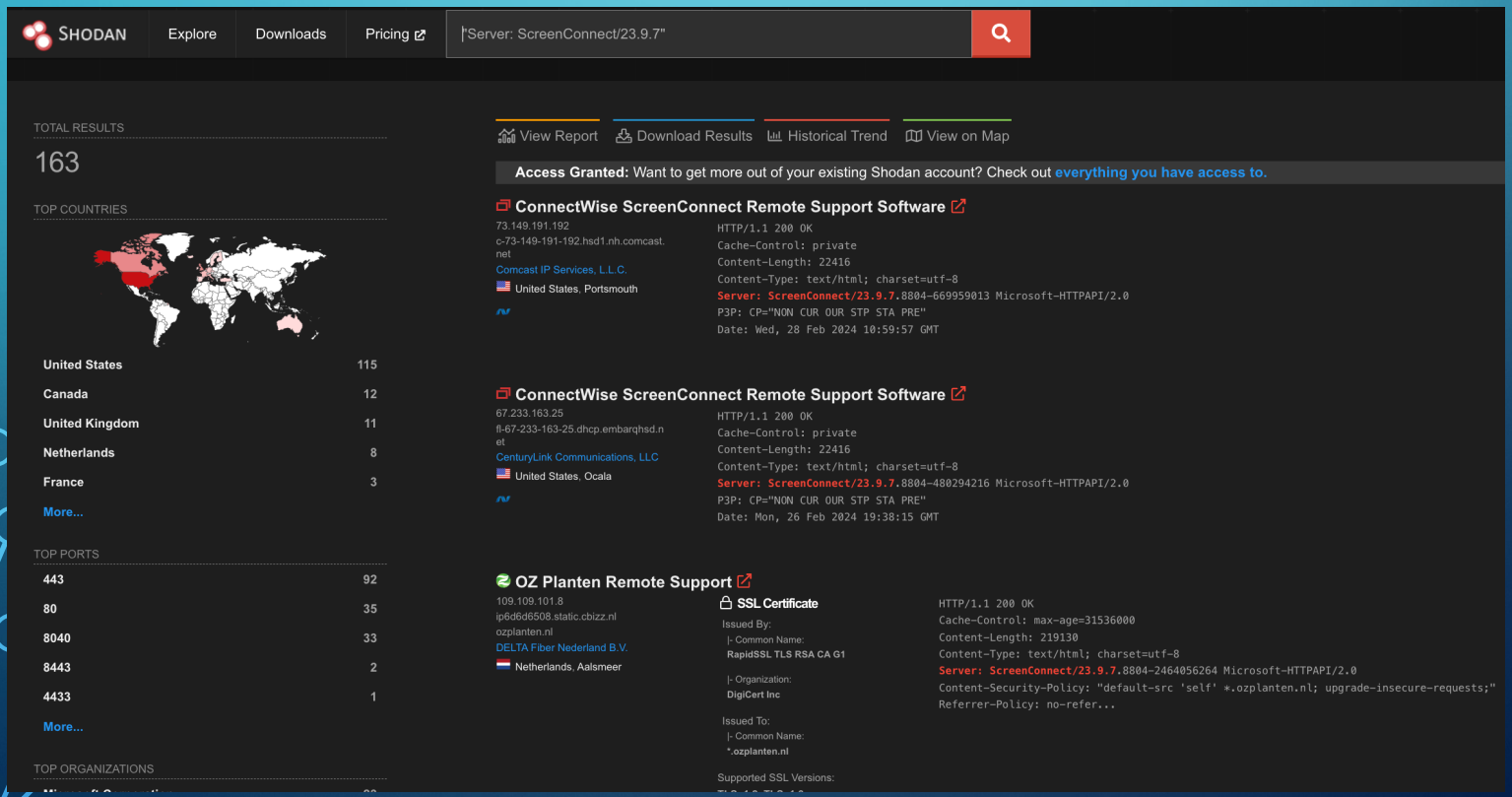
# PATCH LAG – IVANTI EPMM (MOBILE IRON)

- Authentication bypass vulnerability
- Ivanti EPMM 11.10 and older (has a particular favicon)



# PATCH LAG – SCREEN CONNECT

- Authentication Bypass Using an Alternate Path or Channel vulnerability; Path-traversal vulnerability
- ScreenConnect 23.9.7 and older



# PATCH LAG

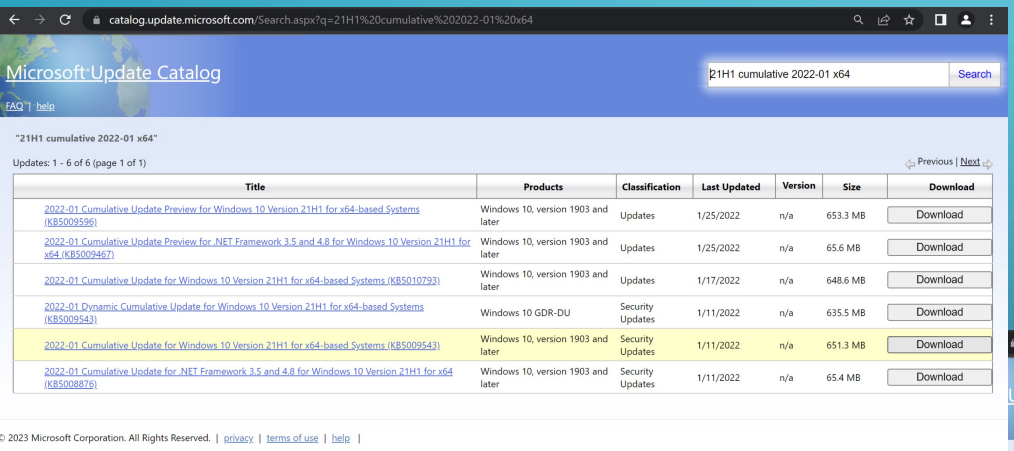
- Systems doesn't have to sit unpatched for months to get exploited
- UHC's Change Healthcare org was ransomwared last week, and the attackers used the Screen Connect vulnerability to get the access
  - The Screen Connect vulnerability is less than a month old
  - Many organizations take 2-3 months to patch

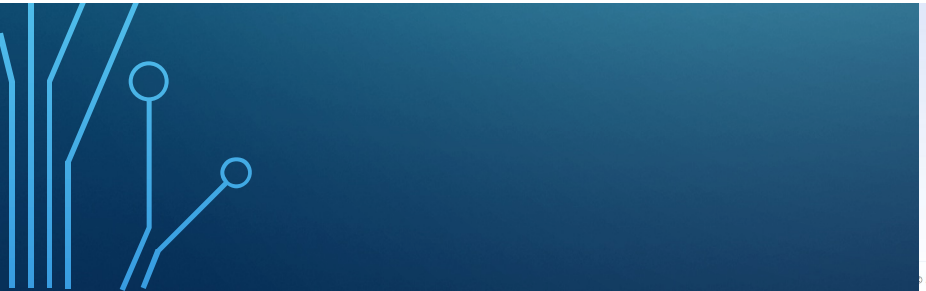
# WINDOWS PATCHES

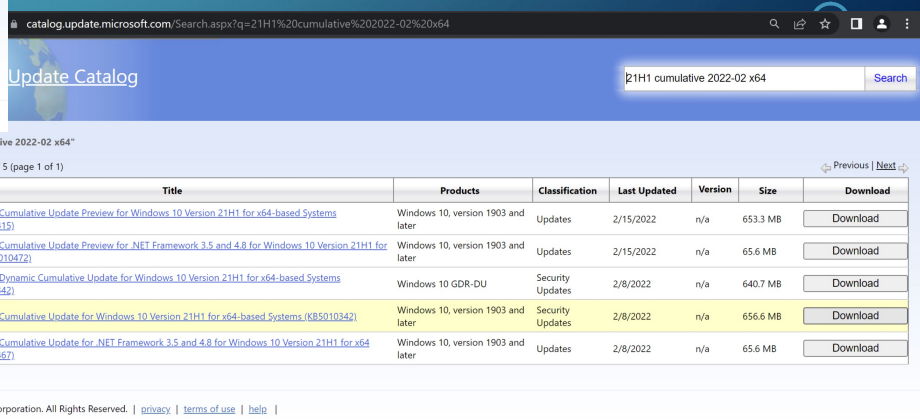
- Windows patches are some of the most targeted for 1-day exploits, because so many computers run Windows
  - Especially business computers, which usually contain very sensitive data
- However, Windows patch binaries come in rollups, and extracting the binaries can be a complicated process, involving several steps

# EXTRACTING WINDOWS PATCHING

- You can obtain the Windows Patch files from the Microsoft Update Catalog





# EXTRACTING WINDOWS PATCHING

- The patches are packed, and need to be unpacked

```
PS C:\Users\student> expand.exe -F:* .\windows10.0-kb5009543-x64_c7b660efcaaa2dd1d55fe91c5cea3d9b209a15cd.msu msu
PS C:\Users\student> expand.exe -F:* .\windows10.0-kb5010342-x64_f865479b6847db1aab8d436a37a964f31c853887.msu msu
PS C:\Users\student> expand.exe -F:* .\Windows10.0-KB5009543-x64.cab cab1
PS C:\Users\student> expand.exe -F:* .\Windows10.0-KB5010342-x64.cab cab2
PS C:\Users\student> expand.exe -F:* .\Windows10.0-KB5009543-x64.cab cabA
PS C:\Users\student> expand.exe -F:* .\Windows10.0-KB5010342-x64.cab cabB
```

# EXTRACTING WINDOWS PATCHING

```
PS C:\Users\student\Desktop\patchDiffing\msu\cab1> ls


    Directory: C:\Users\student\Desktop\patchDiffing\msu\cab1


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/19/2023   9:50 AM                cabA
------          1/8/2022   3:27 AM       15318363 SSU-19041.1371-x64.cab
------          1/8/2022   3:28 AM            202 toc.xml
------          1/8/2022   3:28 AM           8819 update.cat
------          1/8/2022   3:28 AM         421548 update.mum
------          1/8/2022   3:27 AM      650697678 Windows10.0-KB5009543-x64.cab
```

```
PS C:\Users\student\Desktop\patchDiffing\msu\cab1\cabA> ls


    Directory: C:\Users\student\Desktop\patchDiffing\msu\cab1\cabA


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/19/2023  10:08 AM                update
------          1/7/2022  10:12 PM             93 cabinet.cablist.ini
------          1/7/2022  10:12 PM       98808009 Cab_1_for_KB5009543_PSFX.cab
------          1/7/2022  10:08 PM      563004122 Cab_2_for_KB5009543_PSFX.cab
------          1/7/2022   9:25 PM           8819 update.cat
------          1/7/2022   9:23 PM         421548 update.mum
```

# EXTRACTING WINDOWS PATCHING

- Since Windows patches are cumulative (they contain all of the fixes from previous patches as well), we need to extract the delta (change) between the latest patch and the binary that is currently on our computer
  - You can use delta_patch.py, a delta extraction script written by Jaime Geiger
  - https://gist.github.com/wumb0/9542469e3915953f7ae02d63998d2553

# EXTRACTING WINDOWS PATCHING

- First, locate the original version of the binary on Windows

```
PS C:\Users\student\Desktop\patchDiffing\msu\cab1\cabA> gci -rec C:\Windows\WinSxS\ -Filter ntoskrnl.exe


    Directory:
    C:\Windows\WinSxS\amd64_microsoft-windows-os-kernel_31bf3856ad364e35_10.0.19041.1566_none_e0d5c562696cbc64


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2/17/2022  11:43 AM       10849616 ntoskrnl.exe


    Directory:
    C:\Windows\WinSxS\amd64_microsoft-windows-os-kernel_31bf3856ad364e35_10.0.19041.1566_none_e0d5c562696cbc64\f


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2/12/2022   1:44 AM        1574157 ntoskrnl.exe


    Directory:
    C:\Windows\WinSxS\amd64_microsoft-windows-os-kernel_31bf3856ad364e35_10.0.19041.1566_none_e0d5c562696cbc64\r


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         2/12/2022   1:44 AM        1340897 ntoskrnl.exe
```

# EXTRACTING WINDOWS PATCHING

- Then run the delta_patch.py script with the current binary, and the delta binary that you extracted from the patch rollup
- This will give you the binary with the added delta patched content

```
PS C:\Users\student> python C:\lab\day4\delta_patch.py -i ntoskrnl.exe -o ntoskrnl_2022-01.exe C:\Users\student\Desktop\
patchDiffing\r\ntoskrnl.exe C:\Users\student\Desktop\patchDiffing\msu\cab1\cabA\update\amd64_microsoft-windows-os-kernel
_31bf3856ad364e35_10.0.19041.1466_none_e0e0954e6964a073\f\ntoskrnl.exe
```

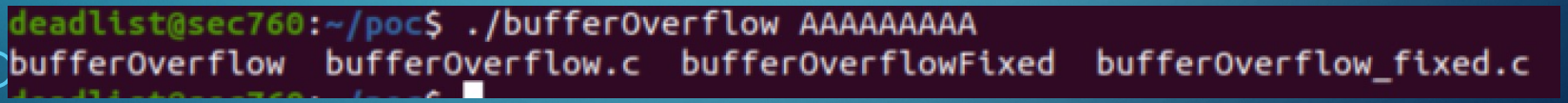
```
PS C:\Users\student> python C:\lab\day4\delta_patch.py -i ntoskrnl.exe -o ntoskrnl_2022-02.exe C:\Users\student\Desktop\
patchDiffing\r\ntoskrnl.exe C:\Users\student\Desktop\patchDiffing\msu\cab2\cabB\update\amd64_microsoft-windows-os-kernel
_31bf3856ad364e35_10.0.19041.1526_none_e0d1c43a69705708\f\ntoskrnl.exe
```

# EXPLOIT DEVELOPMENT

- Note that not all vulnerabilities are the same level of easiness/difficulty to exploit
  - Some are easy, like a filter that misses removing a special character in an encoded format
    - Just send an input with that special character encoded in the missed format
  - Others are very difficult to exploit
    - A memory region that needs to be groomed via multiple inputs, in order to get the needed values into memory at the needed place

# POC

- I wrote a simple program with a buffer overflow
  - Since trying to exploit a patched Windows vulnerability can be too long and complex for a presentation
- The program accepts user input, and normally runs the "ls" command, which prints the directory



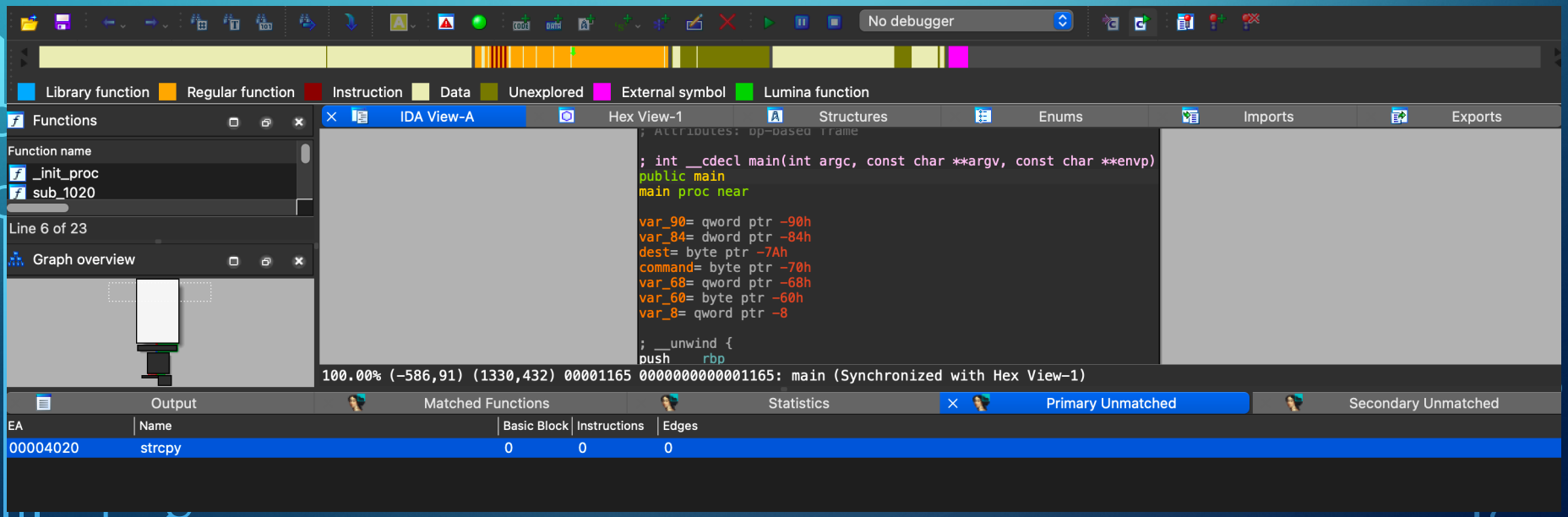
# POC

- But when your input overflows into the “ls” string’s memory, you can replace the “ls” command with anything you want
  - Print the /etc/shadow file

```
deadlist@sec760:~/poc$ ./bufferOverflow AAAAAAAAAA"sudo cat /etc/shadow"
root:$6$Pw/srH4fD7dsHnRm$5.yQsUvPLSqGWMdv4S9G/o2j3Z2midj582/gH148CLNiZsRu9py3hgj
/XyvRQVGZJsegkcyrIRlFxegiZTLVs1:19454:0:99999:7:::
daemon:*:18002:0:99999:7:::
bin:*:18002:0:99999:7:::
sys:*:18002:0:99999:7:::
sync:*:18002:0:99999:7:::
games:*:18002:0:99999:7:::
man:*:18002:0:99999:7:::
lp:*:18002:0:99999:7:::
mail:*:18002:0:99999:7:::
```

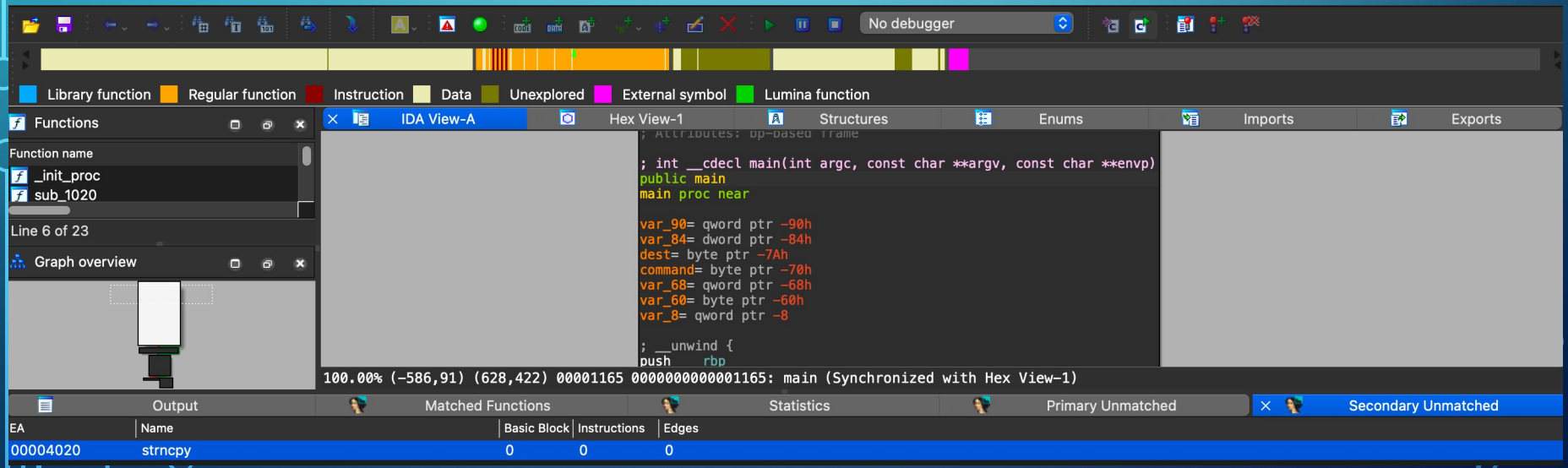
# POC

- BinDiff shows the strcpy function is only used in the vulnerable version of the program
  - The strcpy function doesn't do input length checking, allowing the buffer overflow into the "ls" memory



# POC

- However, BinDiff shows the strncpy function is used in the patched version of the program
  - The strncpy function does do input length checking
  - This prevents long user input from overflowing into the ”ls” memory



# POC

```c
//Poc c program that is vulnerable to a buffer overflow from the buffer variable into the
//modified variable; and then executes the contents of the modified variable.
//The buffer overflow allows you to change the modified variable from "ls" to another
//command.
//*note: to run a command that has spaces, you need to enclose it in quotes
//AAAAAAAAAA"cat /etc/pass"
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
    char modified[100] = "";
    strcat(modified, "ls");
    char buffer[10];

    if(argc == 1) {
        printf(1, "please specify an argument\n");
    }

    strcpy(buffer, argv[1]);
    system(modified);
}
```

# POC

```c
//Poc c program that is vulnerable to a buffer overflow from the buffer variable into the
//modified variable; and then executes the contents of the modified variable.
//The buffer overflow allows you to change the modified variable from "ls" to another
//command.
//*note: to run a command that has spaces, you need to enclose it in quotes
//AAAAAAAAAA"cat /etc/pass"
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
    char modified[100] = "";
    strcat(modified, "ls");
    char buffer[10];

    if(argc == 1) {
        printf(1, "please specify an argument\n");
    }

    strncpy(buffer, argv[1], 10);
    system(modified);
}
```

# POC

- The fixed version of the program doesn't print the whoami command results when you overflow the buffer

```
lab@lab:~/Desktop/patchDiff_poc$ ./bufferOverflow AAAAAAAAAAwhoami
lab
```

```
lab@lab:~/Desktop/patchDiff_poc$ ./bufferOverflow_fixed AAAAAAAAA
bufferOverflow  bufferOverflow.c  bufferOverflow_fixed  bufferOverflow_fixed.c
lab@lab:~/Desktop/patchDiff_poc$ ./bufferOverflow_fixed AAAAAAAAAAwhoami
bufferOverflow  bufferOverflow.c  bufferOverflow_fixed  bufferOverflow_fixed.c
```

You can find these slides, as well as my past presentations at my Github:

https://github.com/koyoresearch/presentations

# TEASER FOR NEXT YEAR – ZERO DAY BUG HUNTING

- Open a random binary in a disassembler, and look for dangerous functions, that don't do bounds checking when handling data
  - strcpy(), strcat(), sprintf(), gets(), etc
- Then walk up the code chain to see if the dangerous function receives user inputted data
- Inspired by a talk that Stephen Sims gave at Sans Hackfest a couple of years ago on how to look for zero days
  - I was like "Wow, it can be that easy?"

# ANY QUESTIONS?